Середовище програмування
IDE (Integrated Development Environment)

Довідка про версії Visual Basic

Visual Basic для Windows з'явився близько 15 років тому. Дебют відбувся 20 березня 1991 року на виставці «Windows World». До цього була версія Quick Basic, але вона тільки по назві схожа на Visual Basic.

VB1 і VB2 - добре були налаштовані на створення прототипів програм і демонстраційних застосувань (не більше).

VB3 – з'явилися засоби роботи з базами даних.

VB4 - з'явилися базові можливості для створення об'єктів, та відповідно - базові засоби об'єктно-орієнтов. програмування.

VB5 і VB6 – аспекти ООП розширені, та сама мова поступово втрачала цілісність, оскільки об'єктно-орієнтовані засоби базувалися на підгрунті, у якому їх підтримка не передбачалася.

Довідка про версії Visual Basic

VB.NET, на відміну від попередніх версій VB, не обмежується додатками, що орієнтуються на графічний інтерфейс, та дозволяє створювати додатки інших типів.

Наприклад, **web-додатки**, **серверні додатки** і навіть **консольні додатки**, що працюють у вікні, схожому на вікно DOS-сеансу.

Тепер у єдиному середовищі працюють **VC++, VJ++, Visual InterDev та Visual Basic**.

Середовище VS.NET можна налагодити так, щоб воно було схоже на IDE від VB6 або іншу IDE по вашому вибору.

Структура програми у VB.NET

В VB.NET кожен проект є частиною того, що Microsoft називає рішенням (solution). Будь-який код, створений в VB.NET IDE, належить до деякого рішення.

Рішення можна розглядати як сховище всієї інформації, необхідної для компіляції програми і її перекладу у форму, придатну для виконання. Таким чином, рішення складається з одного або декількох проектів; різних допоміжних файлів (графічних зображень, ресурсних файлів, метаданих, тобто даних, що описують інші дані, і т.д.); документації у форматі XML і практично всього, що спадає на думку.

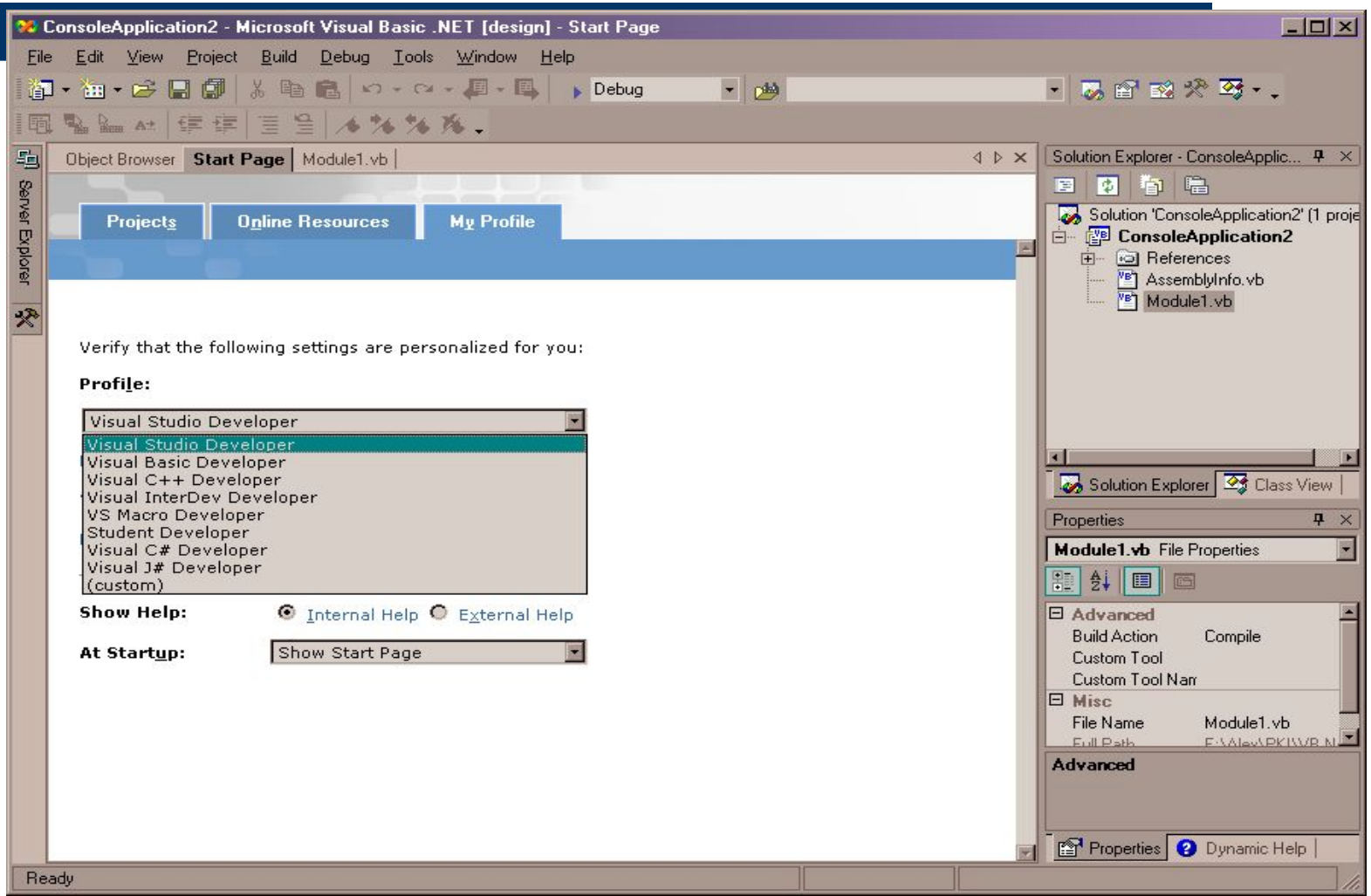
При написанні дрібних програм рішення занадто громіздкі. Але для великих проектів рішення помітно спрощують роботу. Справа в тому, що рішення дозволяє легко вибрати файли, задіяні в вирішенні конкретної проблеми.

Створення нового рішення

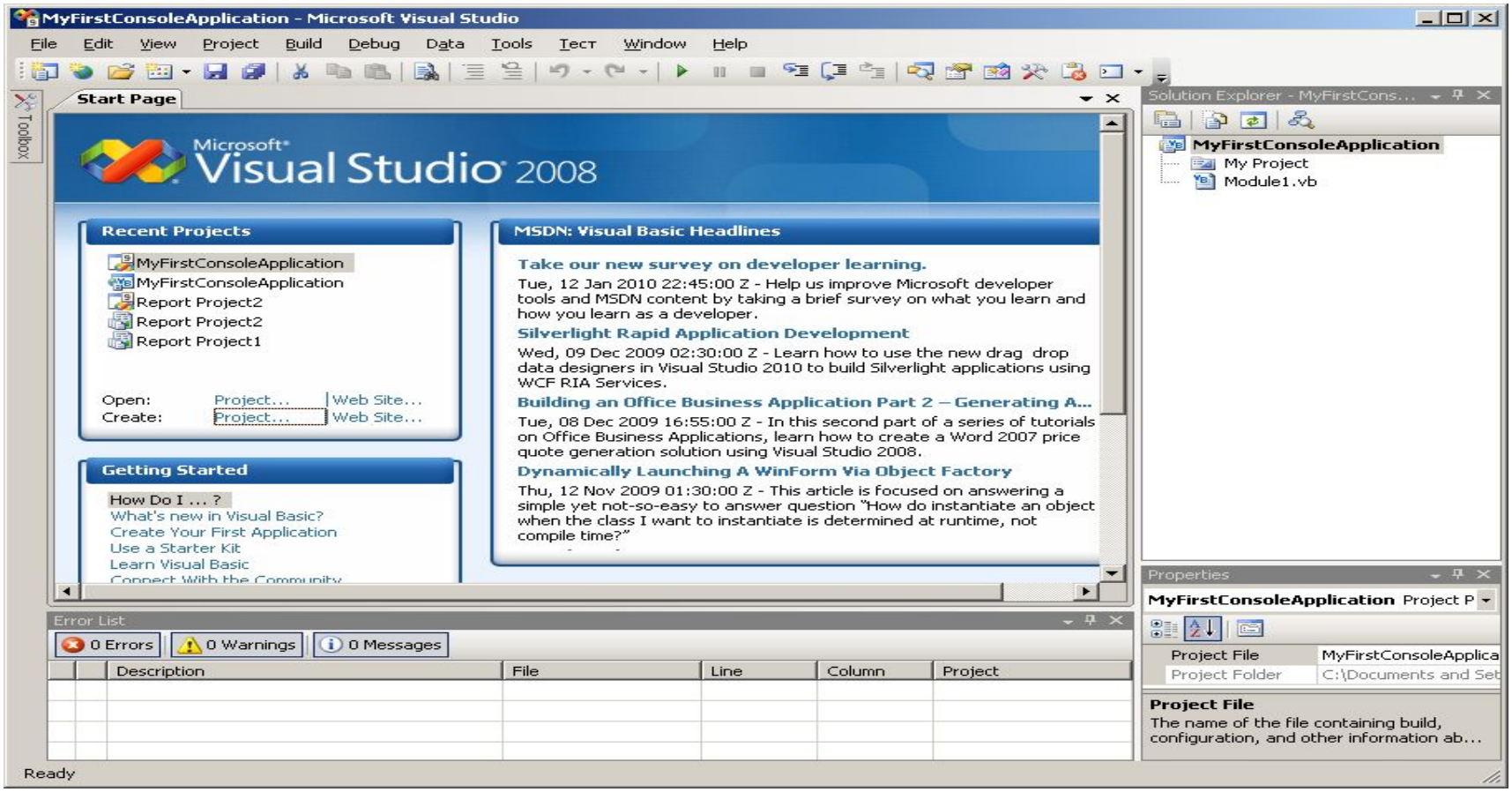
Запускаємо Visual Studio.Net. Заходимо в меню **File > New > Project**. З'явиться віконце **New Project**, в якому обираємо Visual Basic Projects и Console Appication.

Нове рішення створюється командою **File > New**. Вам пропонується вибрати один із двох варіантів: створити новий проект (**New Project**) або порожнє рішення (**Blank Solution**).

Початкова сторінка Visual Studio



Діалогове вікно New Project



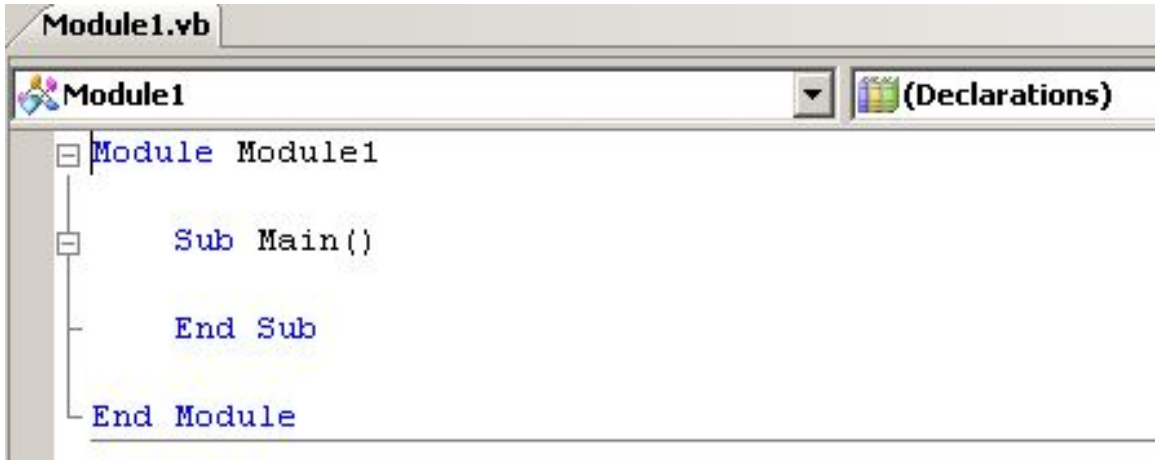
Діалогове вікно New Project

Щоб зосередити всю увагу на можливостях мови VB.NET, не відволікаючись на тонкощі роботи графічних додатків **на початку курсу будемо розглядати тільки консольні додатки**.

Вони працюють у текстовому режимі; з деяким спрощенням можна вважати, що все введення/виведення здійснюється у вікні DOS (дані читаються зі стандартного вхідного потоку й записуються в стандартний вихідний потік).

Текст першої програми

```
Module Module1
    Sub Main()
        System.Console.WriteLine("This is My First App")
    End Sub
End Module
```



Результат виконання першої програми

Запускаємо програму, натиснувши Ctrl+F5. Отримаємо приблизно такий результат:

Для запуска програми можна натиснути і просто F5, але у цьому випадку програма зразу закриється, і побачити результат майже неможливо.

Структура програми VB.NET

Кожний додаток VB.NET повинен мати **точку входу**. У точці входу вміщується код, який автоматично виконується при запуску, після чого керування передається іншому коду програми.

У відносно простих графічних додатках точка входу може асоціюватися з початковою формою. Але, код форм Windows досить складний і пошук точки входу може викликати певні утруднення.

VB.NET дозволяє легко створювати традиційні консольні додатки, які часто застосовуються при програмуванні серверних сценаріїв.

Структура програми VB.NET

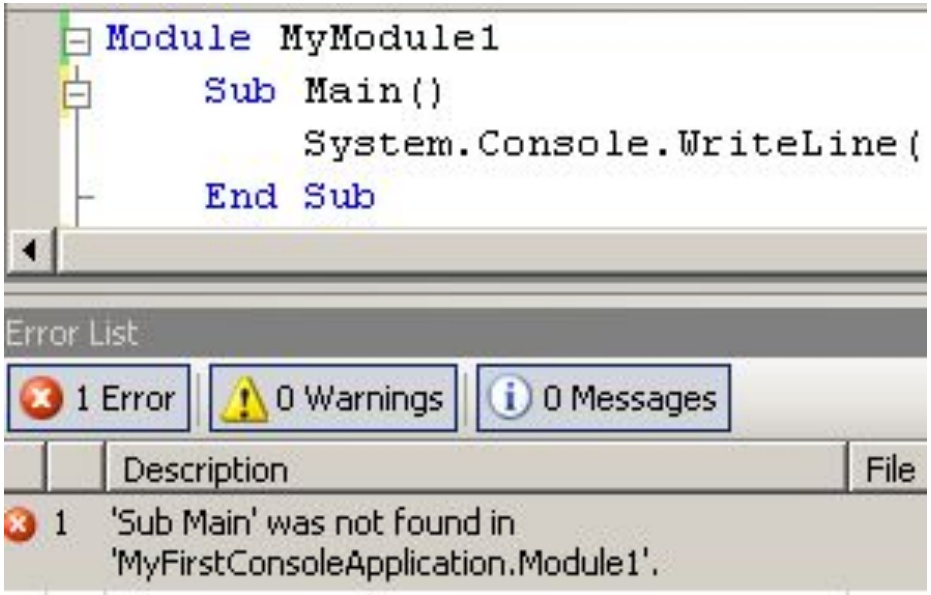
Точкою входу консольного додатка є процедура **Sub Main** модуля. Якщо вибрати в діалоговому вікні New Project (слайд 7) значок консольного додатка (Console Application), VB.NET автоматично генерує «кістяк» додатка із точкою входу - процедурою **Sub Main**:

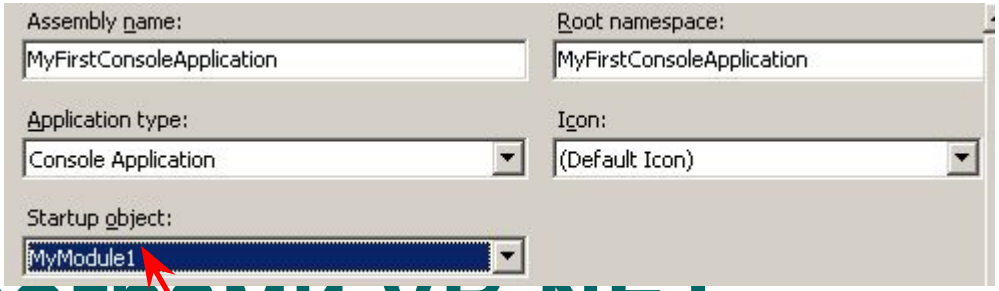
```
Module Module1
    Sub Main()
    End Sub
End Module
```

Структура програми VB.NET

У прикладі використане ім'я Module1, прийняте за замовчуванням.

За правилами ім'я модуля повинне збігатися з ім'ям файлу. Якщо ви змінили ім'я модуля в першому рядку: MyModule1, то при спробі запустити консольний додаток виводиться повідомлення про помилку:





Структура програми VB.NET

Перейменування модуля після його створення виконується в такий спосіб:

1. Змінити ім'я модуля у вікні програми.
2. Змінити ім'я файлу модуля у вікні рішення.
3. Клацнути правою кнопкою миші в рядку ConsoleApplication вікна рішення й вибрати у контекстному меню команду Properties (або виконати команду Project > Properties).
4. Переконатися в тому, що в списку Startup object діалогового вікна, що з'явилося обране нове ім'я модуля.

Програма VB .NET (рішення) може складатися з декількох модулів, але наявність процедури Sub Main **допускається тільки в одному модулі**. Додаток завершується по досягненні команди **End Sub** процедури **Sub Main**.

Модифікована перша програма

```
Module Module1
    Sub Main()
      System.Console.WriteLine("This is My First App")
      System.Console.ReadLine()
    End Sub
End Module
```

Включення у програму рядка ReadLine, дає можливість затримати консольне вікно до натискання клавіші Enter (надзвичайно корисний метод ReadLine() описаний на слідуючому слайді).

Пояснення та коментарі до першої програми

У наведених простеньких програмах викликається метод WriteLine класу Console, призначений для виводу тексту з наступним переведенням рядка (в объектно-орієнтованому програмуванні функції класів звичайно називаються методами).

Метод WriteLine належить до числа загальних (shared) методів, вони також називаються методами класу. Загальні методи докладно будемо розглядати пізніше.

У модифіковану версію програми додано виклик методу ReadLine, що очікує натискання клавіші Enter (метод ReadLine зазвичай використовується в правій частині команди присвоювання, щоб уведений з консолі текст був збережений у заданій змінній).

У наведених програмах варто звернути увагу на парі неочевидних обставин. При виклику методу звичайно вказується конкретний екземпляр класу. Виключення із цього правила становлять особливі методи класу, які називаються загальними методами. **Загальні методи існують на рівні класу, а не його окремих екземплярів.**

Круглі дужки при виклику методів **обов'язкові** - звичайно IDE додає їх автоматично, але краще не забувати про це.

Модифікований текст програми

```
Imports System
    Module Module1
     Sub Main()
          Console.WriteLine("This is My First App")
     End Sub
End Module
```

На початку програми ми імпортували до неї простір імен System. Це означає, що для всього, що входить у цей простір імен, ми можемо використовувати скорочену форму запису замість повної.

У програмі є метод (функція) **Main**. Такий метод обов'язково повинен бути у програмі. Саме з нього розпочинається виконання будь-якої програми.

Рядок меню

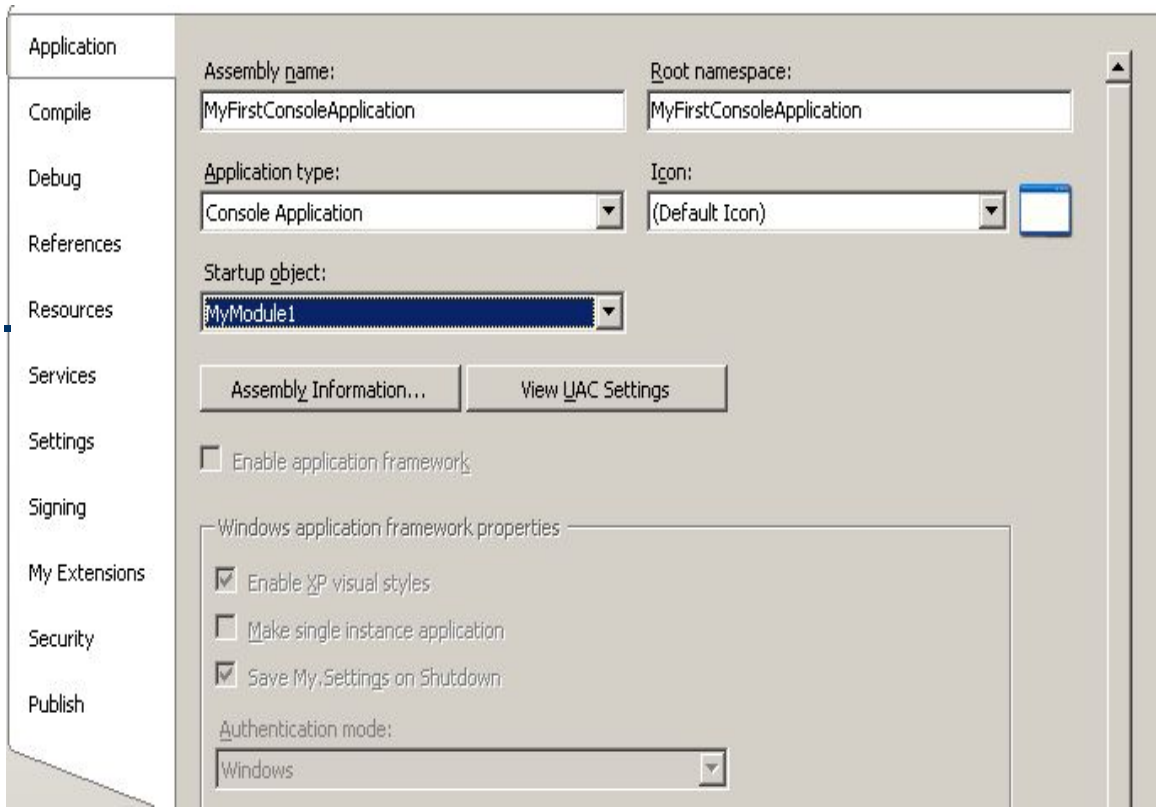
- **File** – команди операцій які пов’язані із файлами – створення, відкриття, збереження, закриття…
- **Edit** – команди операцій редагування файлів – включає видалення, копіювання, переміщення…
- **View** – команди відображення вікон VS.Net.
- **Project** – команди керування проектом, що розробляється.
- **Build** – команди компіляції і компоновки проекту.
- **Debug** – команди налагодження та запуску проекта.
- **Data –** команди роботи із базами даних.
- **Tools** – меню команд налагодження VS.Net.
- **Тест** – команди організації тестування проекту.

Вікно властивостей проекту

Для відображення Property Pages слід виконати команду **Project > ProjectProperties**.

Ім’я скомпільованого файлу

Об’єкт, що запускається

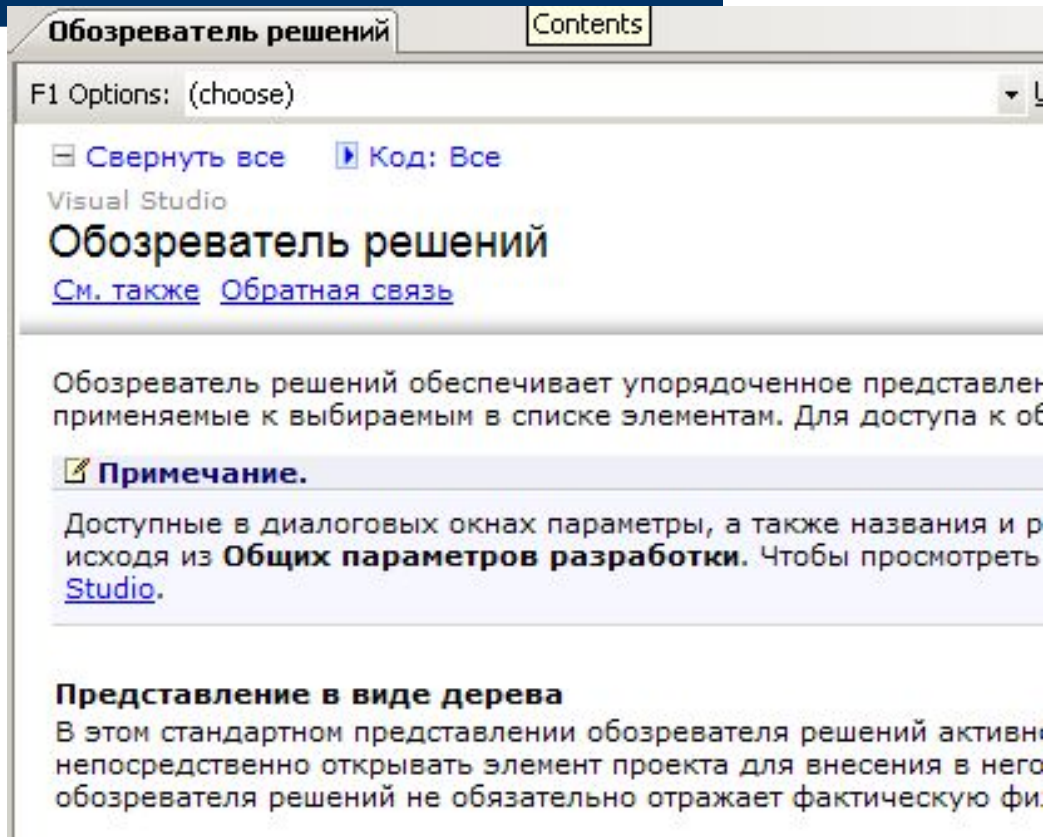


Приклад з введенням/виведенням

```
Imports System
Module Module1
    Sub Main()
        Dim Family As String    ' оголошення текстової змінної
        Dim age As Double       ' оголошення числової змінної
        Console.WriteLine("Введіть прізвище та скільки Вам років")
        Family = Console.ReadLine() ' вводимо прізвище
        age = CInt(Console.ReadLine()) ' вводимо вік
        Console.WriteLine("Прізвище: {0}, прожив {1} років", Family, age)
    End Sub
End Module
```

Довідкова система

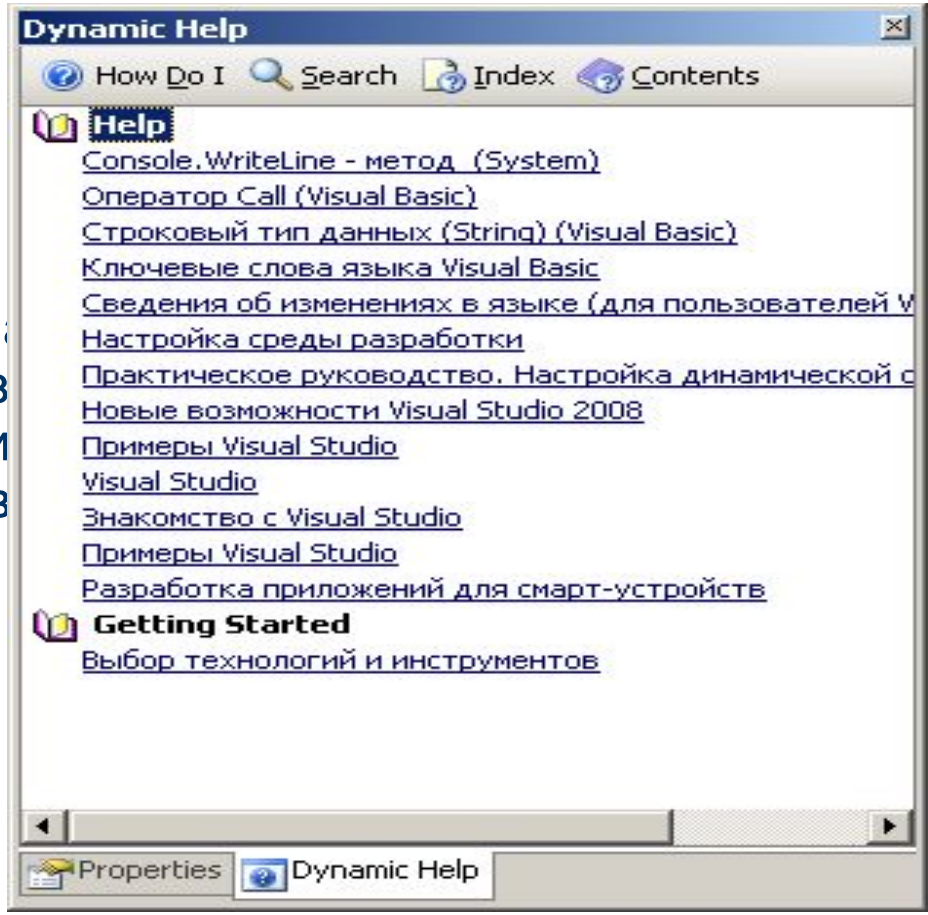
В IDE існує режим контекстної довідки. На малюнку показано вигляд довідки, яку викликали клавішею F1 при знаходженні фокуса у вікні Solution Explorer.



Довідкова система

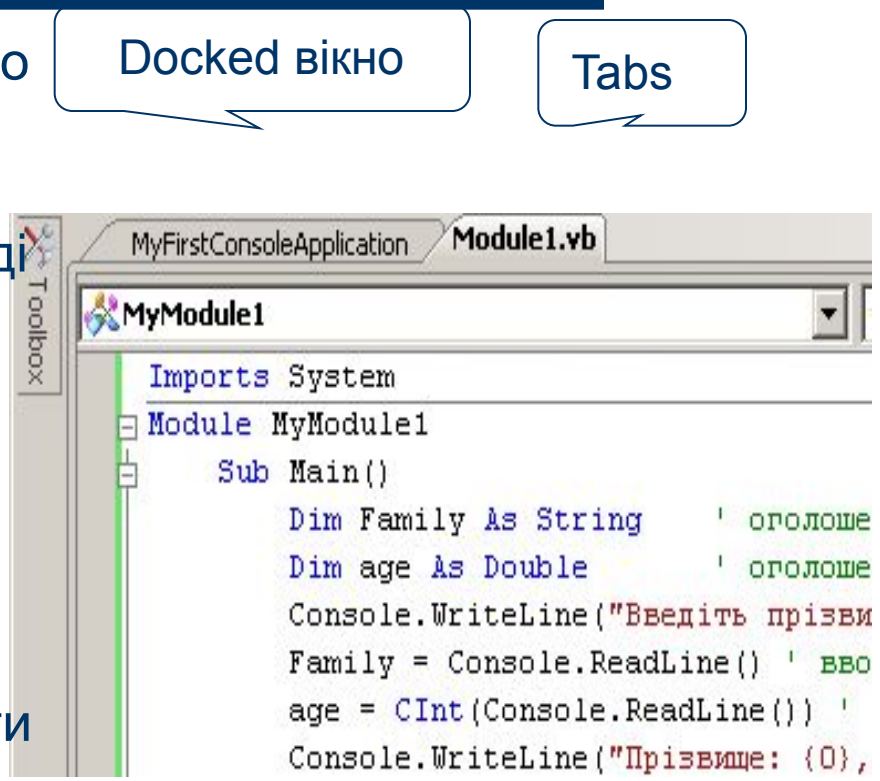
Також підтримується режим динамічної довідки (клавіші **Ctrl+F1**), що автоматично відслідковує ваші дії й намагається викликати відповідний розділ довідки. На малюнку показано список роз динамічної довідки, отримани момент коли курсор знаходив на команді Writeline.

У динамічної довідки є один серйозний недолік - вона інтенсивно витрачає ресурси процесора.



Система відображення вікон в IDE

- Є ефектна особливість IDE - якщо (docked) вікно повністю перекриває інше вікно, то ці вікна відображаються у вигляді корінців (tabs).
- Щоб викликати сховане вікно, досить клацнути на корінці й перетягнути його мишею.
- Щоб змінити порядок вікон (наприклад, для економії місця), слід просто перетягнути один корінець поверх іншого.

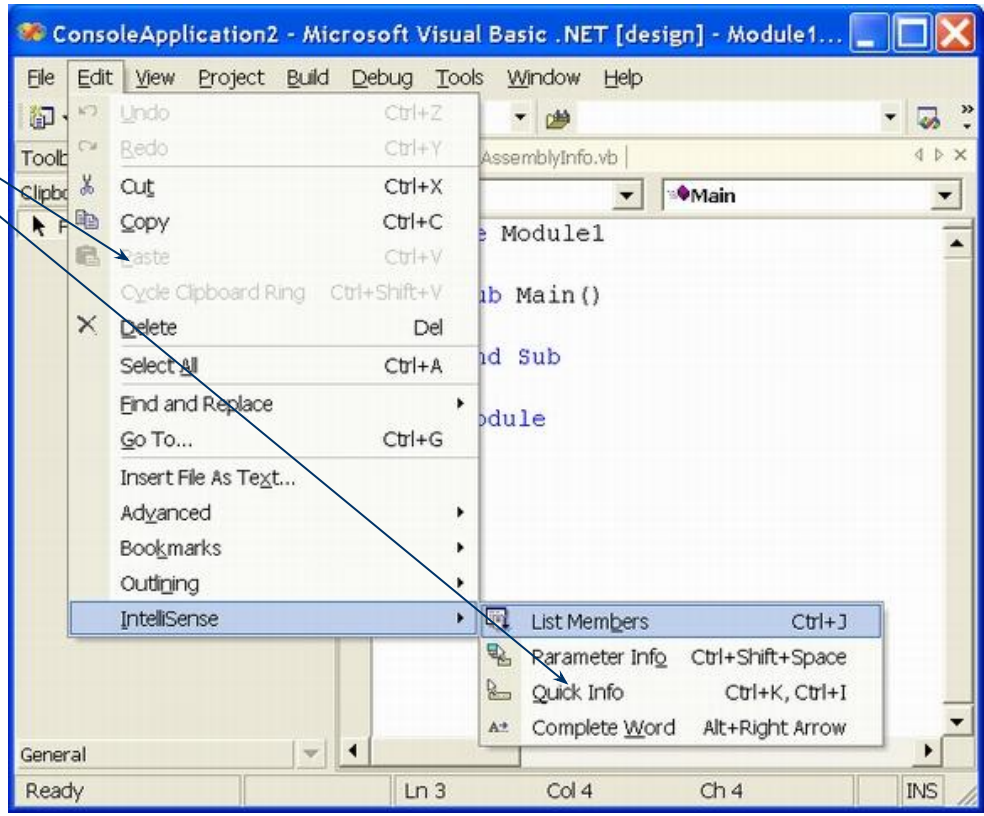


Основні вікна IDE

Контекстні меню викликаються правою кнопкою миші (тут показане контекстне меню редактора).

Кнопки панелей інструментів мають підказки (Hint).

На деяких кнопках є стрілки, що показують, що при натисканні кнопки відкривається меню.



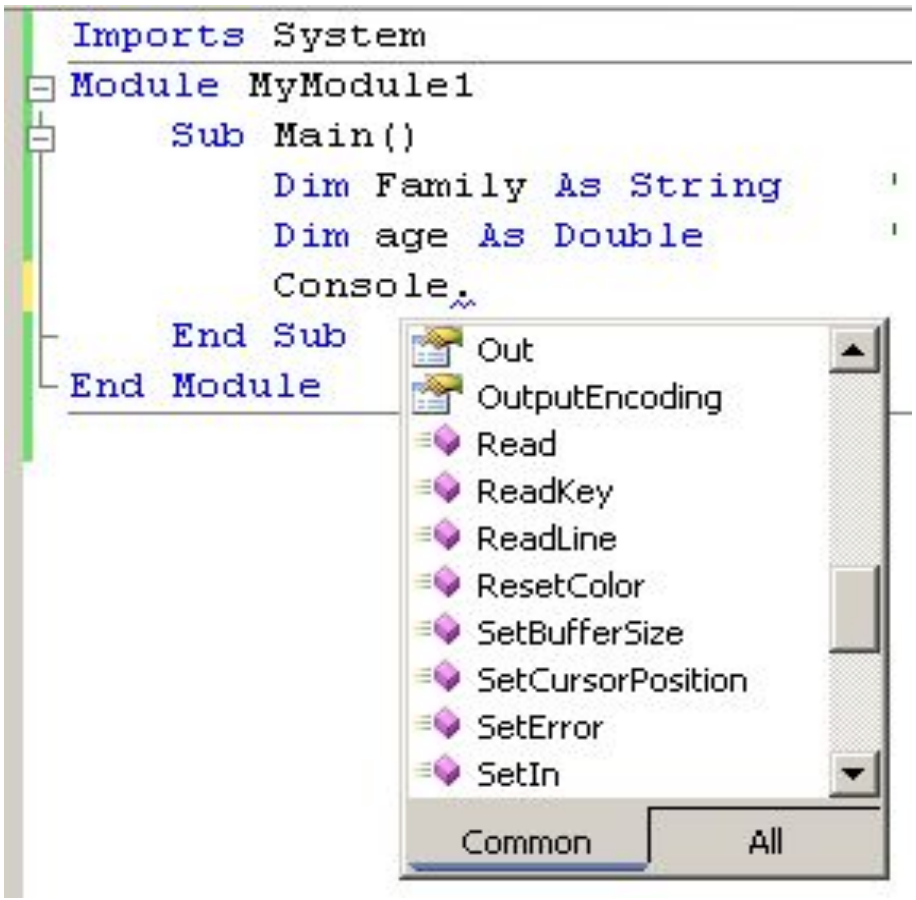
Редактор

Редактор має повний набір стандартних можливостей, який є у редакторах такого роду (вирізання, вставка, пошук/заміна й т.д.). Для роботи з ними можна використовувати стандартні комбінації клавіш Windows (Ctrl+C – копіювати, Ctrl+X - вирізати, Ctrl+V - вставити й т.д.).

Повний список сполучень клавіш викликається з меню Edit; крім того, він наведений у розділі «Editing, shortcut keys» довідкової системи.

Редактор

Є дуже зручний засіб **IntelliSense**, що видає інформацію про методи заданого об'єкта або про параметри, що передані при виклику функції. Звичайно IntelliSense автоматично викликається при введенні символу «.».



Налагодження редактора

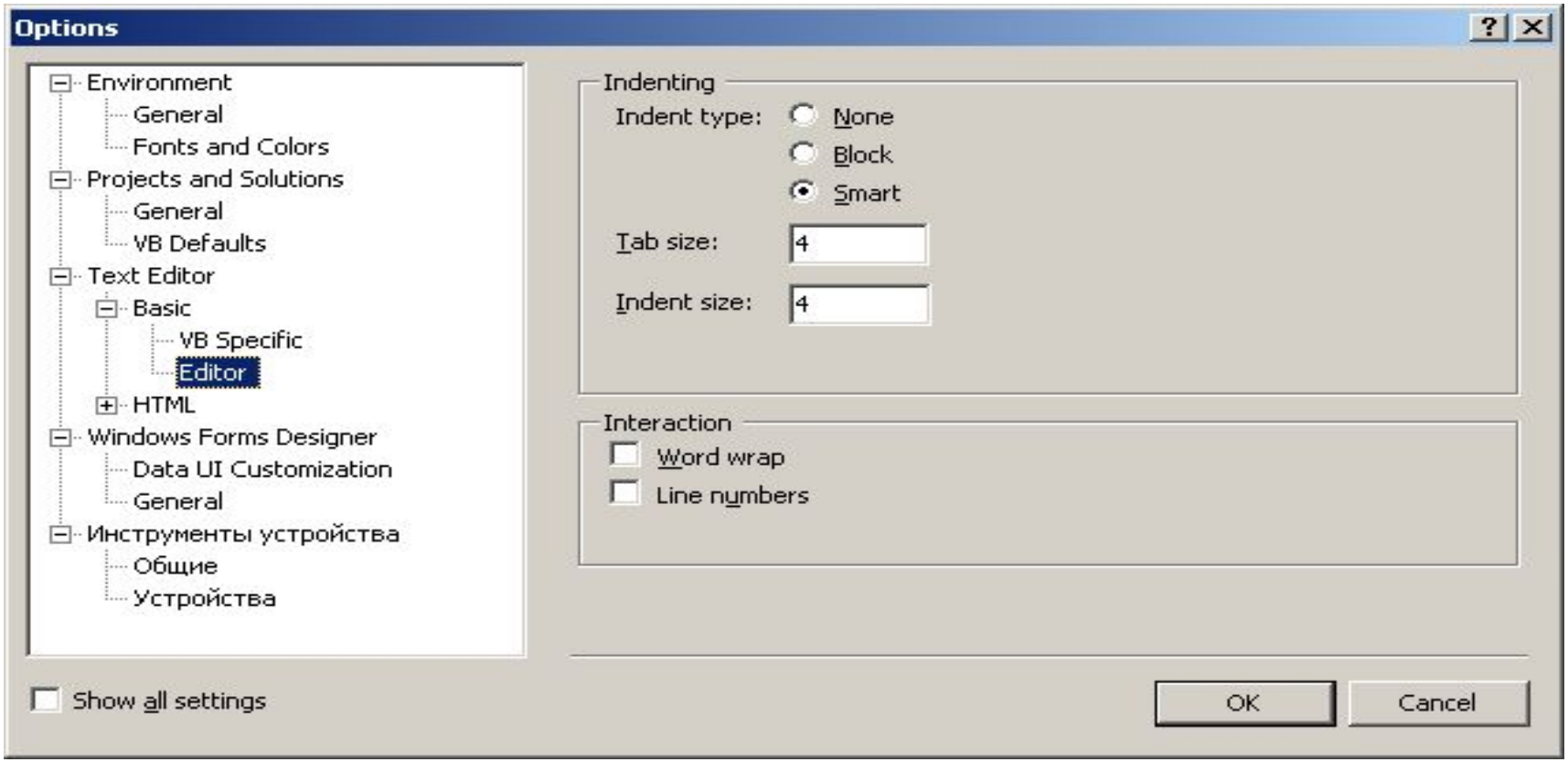
Налагодження більшості глобальних параметрів редактора виконується в діалоговому вікні **Tools > Options > Text Editor** – там 10 закладок.

Наприклад, щоб вибрати **розмір позицій табуляції**, клацніть у рядку Text Editor і виберіть потрібне значення для всіх мов або тільки для VB.

Там же вибирається **режим створення відступів**: **None** (відступи відсутні), **Block** (курсор вирівнюється по початку попереднього рядка) або **Smart** (автоматичне створення відступів у тілі циклу, як того вимагає гарний стиль програмування).

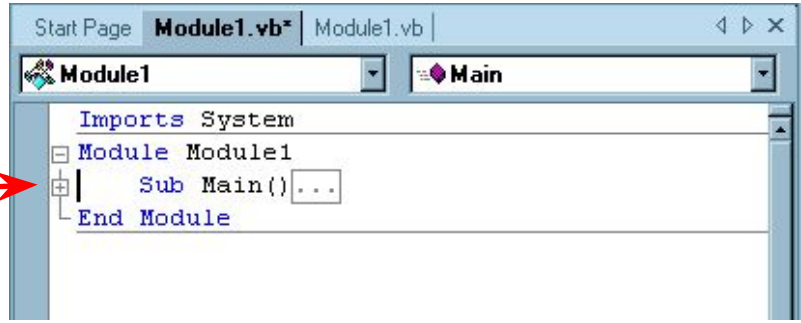
В VB.NET знову стали доступні надзвичайно зручні **команди блокового коментування/зняття коментарів**, що вперше з'явилися в VB5. Тепер ці команди викликаються зі стандартних панелей інструментів IDE .

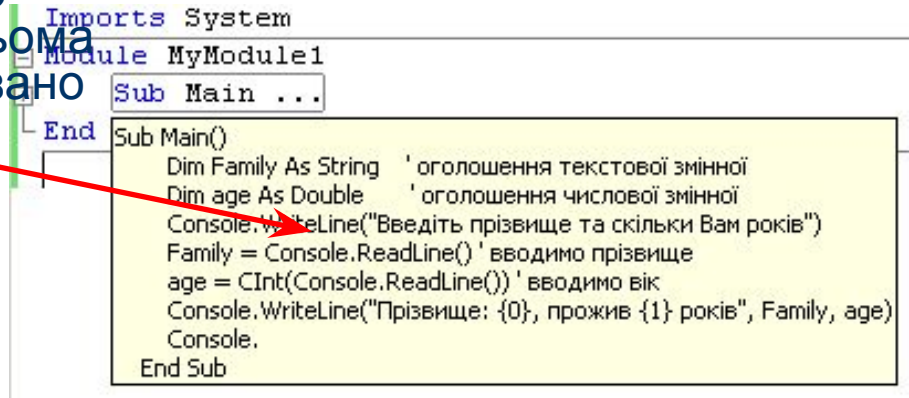
Діалогове вікно Options



Згорнуті області в редакторі

- Редактор підтримує й таку зручну можливість, як **згортка фрагментів програми** й відображення на їхньому місці заголовків.
- Зверніть увагу на значки «+» поруч із деякими рядками на «Згорнуті області в редакторі». Якщо клацнути на такому значку, відкривається відповідна область (region). Якщо затримати покажчик миші над трьома точками (...), на екрані буде показано розгорнутий код.
- Для керування згортанням використовується підменю **Edit > Outlining**.





Багатоелементний буфер обміну

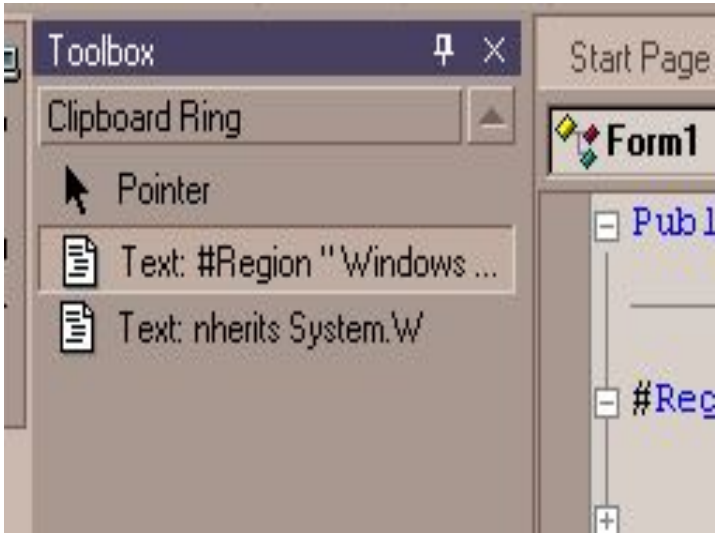
Буфер обміну розраховано на 15 елементів.

Весь вирізаний або скопійований текст передається в багатоелементный буфер обміну, до якого можна звернутися з панелі елементів.

Щоб переглянути поточний вміст буфера, клацніть на корінці **Clipboard Ring** на панелі елементів.

Комбінація клавіш **Ctrl+Shift+V** вставляє черговий елемент буфера в поточний документ.

Багаторазово натискаючи клавіші Ctrl+Shift+V, ви перебираєте вміст буфера. При кожному натисканні Ctrl+Shift+V попередній вставлений фрагмент заміняється поточним елементом буфера.



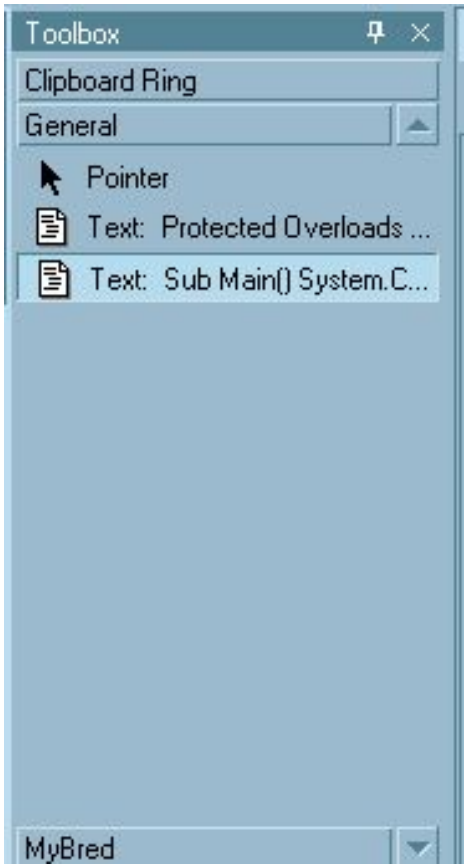
Тимчасове зберігання фрагментів

Будь-який **фрагмент** програмного коду **можна зберегти** для наступного використання на панелі елементів (часто для цієї мети використовують вкладку General, але можна створити нову вкладку - клацніть на панелі правою кнопкою миші й виберіть команду Add Tab з контекстного меню). Така можливість дуже зручна, оскільки в програмах часто зустрічаються фрагменти, що потрібно повторити, а вводити заново занадто довго.

Щоб зберегти фрагмент програми, слід виділити його і перетягнути мишею на панель елементів. Фрагменти залишаються на панелі доти, поки не будуть вилучені за допомогою контекстного меню.

Щоб скористатися збереженим фрагментом, слід перетягнути його мишею в потрібну позицію вікна програми.

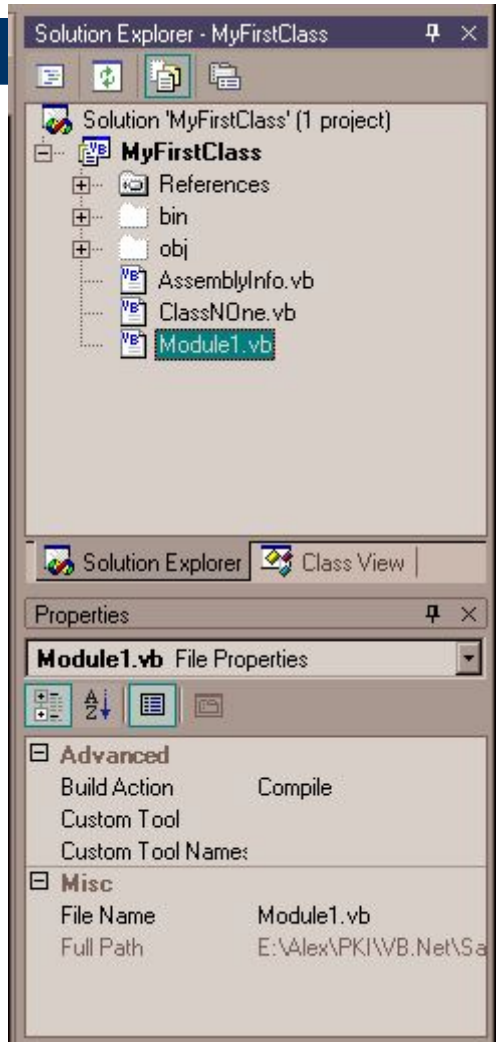
Існує й інший спосіб - виділити позицію вставки й двічі клацнути на збереженому фрагменті.



Вікно рішення (Solution Explorer)

- Вікно Solution Explorer активізується вибором команди View-Solution Explorer або застосуванням інструмента

- У цьому вікні розміщується ієрархічна структура файлів поточного проекту.
- У проекті автоматично створюється по одному файлу для кожної форми, модуля, класу…
- Крім того тут розміщено вузол References з переліком усіх імпортуємих до проекту просторів імен.



Вікно рішення (Solution Explorer)

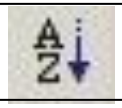
За замовчуванням ім'я рішення збігається з ім'ям першого створеного в ньому проекту. На малюнку зображене рішення MyFirstClass, що містить файл із ім'ям Module1.vb.

В VB.NET всім файлам з кодом Visual Basic незалежно від їхнього типу привласнюється розширення .vb - розширення .frm, .bas і .cls не використовуються.

Втім, одна важлива особливість залишилася незмінною: файли з розширенням .vb, як і в VB6, містять звичайний текст (причому в безкоштовно розповсюджуваний пакет .NET SDK входить автономний компілятор VB для компіляції програм, написаних у зовнішньому текстовому редакторі).

Вікно властивостей (Properties)

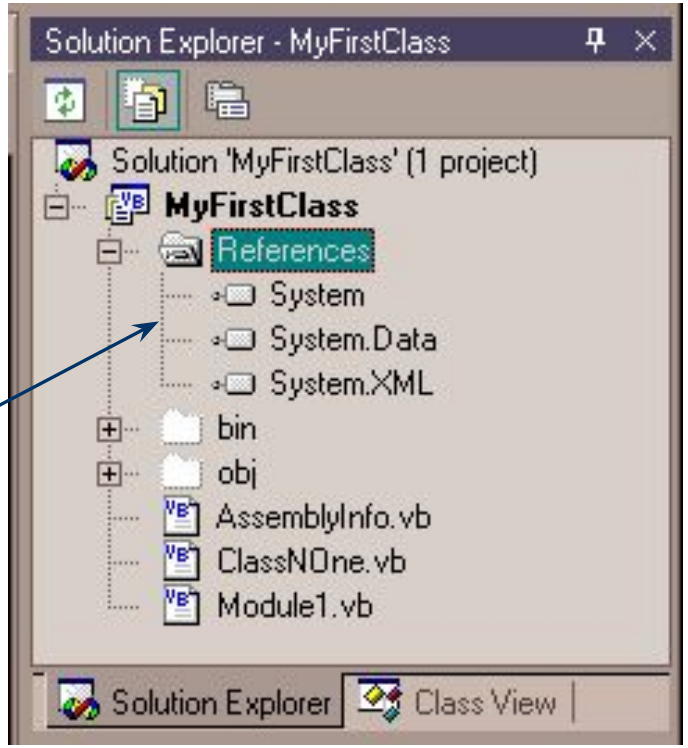
Функції вікна властивостей в VB.NET не обмежується простим завданням властивостей елементів керування. Вміст вікна залежить від того, що в даний момент виділене в IDE. Ім'я й тип виділеного елемента зазначені в списку, що перебуває у верхній частині вікна. Щоб змінити значення властивості, клацніть у правій частині рядка й починайте вводити символи.

Кнопка	Опис
	Виводить алфавітний список всіх властивостей і їхніх значень із розбивкою по категоріях
	Сортує властивості й події за алфавітом
	Виводить список властивостей об'єкта. Якщо з об'єктом зв'язані події, вони також включаються в список
	Відображає сторінку властивостей, якщо вона існує (сторінки властивостей спрощують уведення складних значень)

Вікно зовнішніх посилань (References)

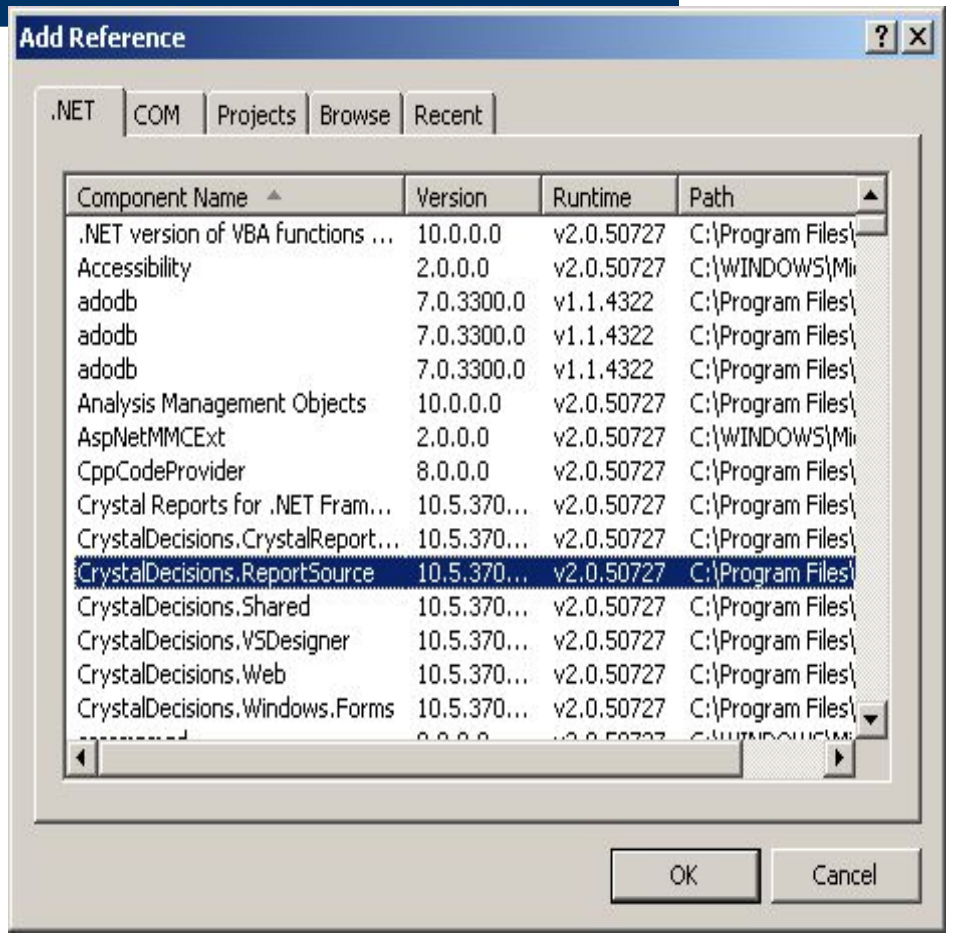
Ієрархічний список файлів у вікні рішення містить розділ References з інформацією про всі зборки (assemblies), які відбулися у проекті.

Є можливість розгорнути розділ дерева, клацнувши на «+». Зверніть увагу - імена всіх базових збірок починаються із префікса System.



Створення нових зовнішніх посилань (Add References)

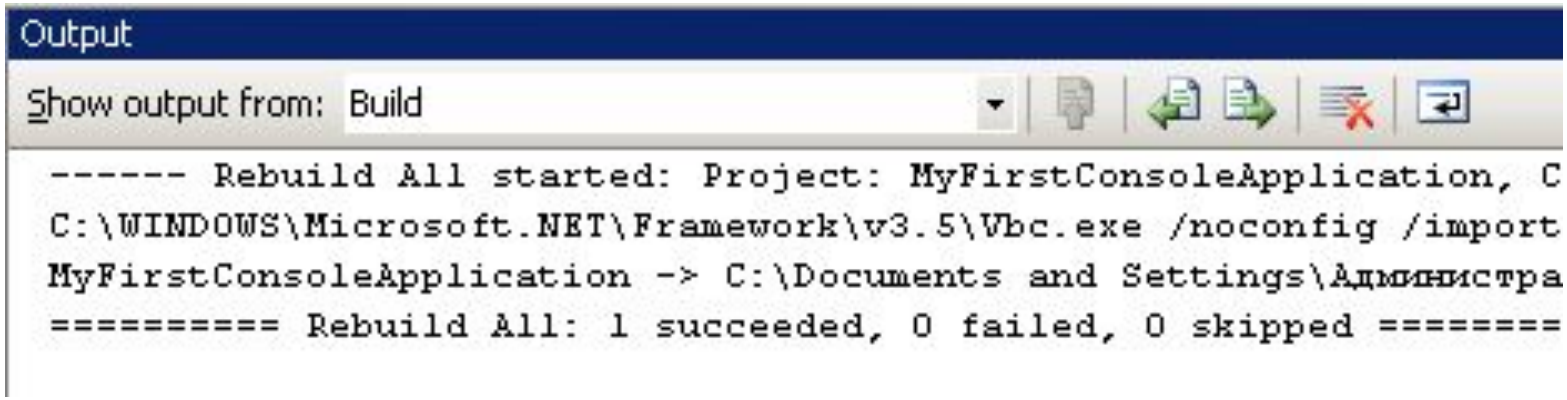
- Щоб добавити до проекту зовнішні посилання можна клацнути правою кнопкою миші в рядку References вікна рішення й вибрати у контекстному меню команду Add Reference
- Також можна скористатися командою Project > Add Reference головного меню).
- На екрані з'являється діалогове вікно, У якому можна додавати зовнішні посилання трьох типів: .NET, COM і інші проекти (Projects).



Вікно виводу (Output)

У вікні виводу Output (викликається командою View > Other Windows або комбінацією клавіш Ctrl+Alt+O) відображається поточна інформація про проект.

При побудові рішення у цьому вікні компілятор виводить повідомлення як про успішне завершення, так і про помилки що виникли.



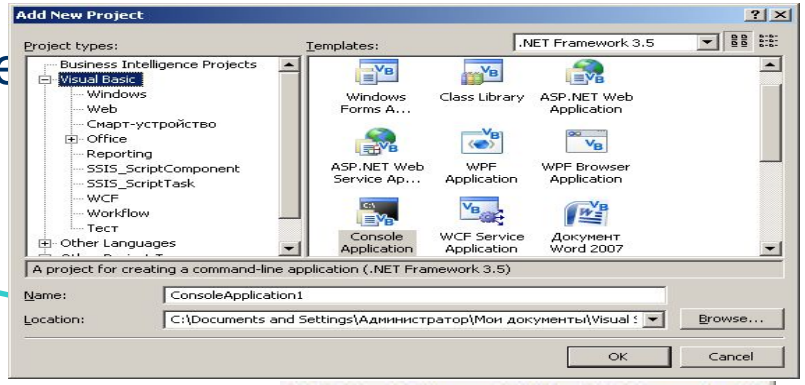
Вікно команд (Command Window)

Викликається командою View > Other Windows або комбінацією клавіш Ctrl+Alt+A.
Не підтримує IntelliSense і не працює в режимі конструювання.
Вікно команд дозволяє взаємодіяти із середовищем IDE.
Вікно команд працює у двох режимах: у режимі команд (Command) і в режимі безпосереднього уведення (Immediate).
Для перемикання між режимами у вікні вводяться рядки >cmd або immed (без префікса <<>»!). Нижче перераховані комбінації клавіш, які використовуються при переміщенні у вікні команд.

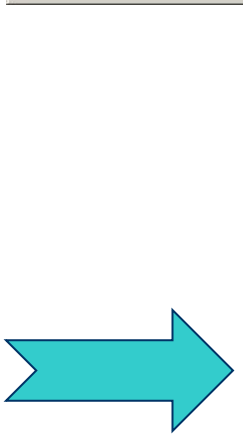
Операція	**Клавіші**
Перебір раніше уведених команд	нагору, униз
Прокручування вікна нагору	Ctrl + нагору
Прокручування вікна вниз	Ctrl + униз

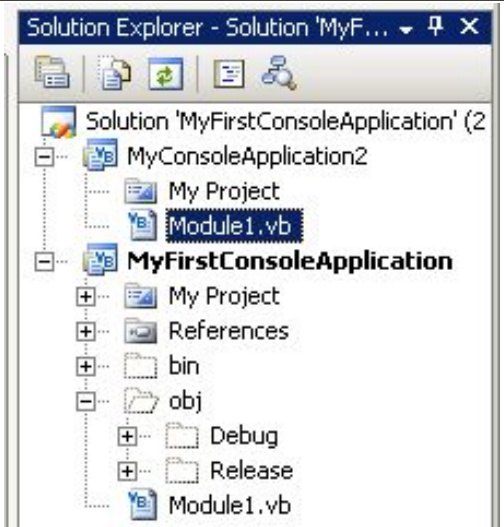
Включення нових проектів у рішення

Слід виконати команду **File > Add > New Project,** з'являється знайоме вікно New Project. Новий проект створюється в новому рішенні.



Тут до рішення добавлено іще один проект MyConsoleApplication2 (раніше був тільки проект MyFirstConsoleApplication). Ці проекти можуть взаємодіяти між собою.

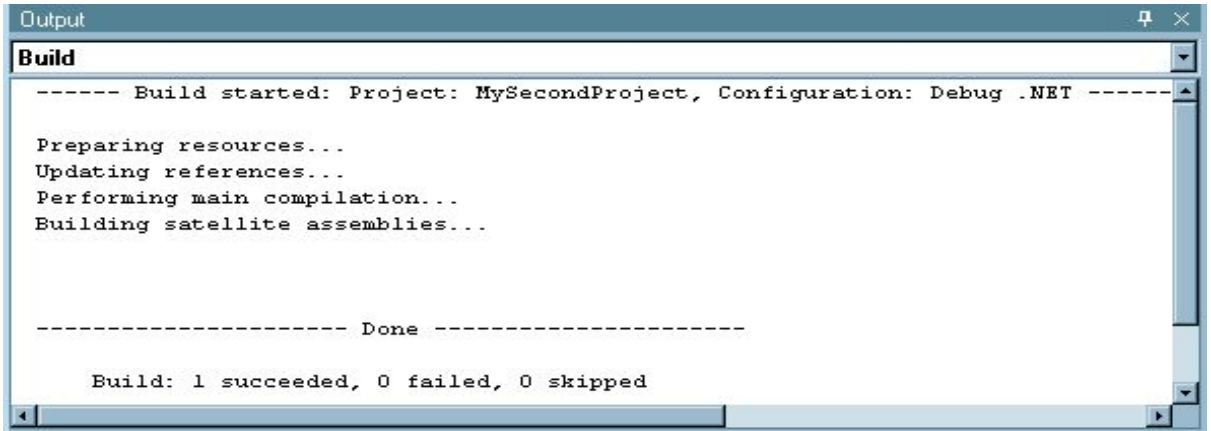




Компіляція

У рішення входить два проекти. Файл-результат може бути побудований на будь-якому з проектів. Компілювати ці проекти також можна незалежно один від іншого.

Щоб скомпілювати який-небудь проект із поточного рішення, слід у вікні рішення (Solution Explorer) клацнути ПКМ на одному з проектів й вибрати команду Build або Rebuild у контекстному меню.

Якщо Build - компілятор обмежується побудовою частин проекту, що змінилися з моменту останньої побудови. Build використовується частіше, оскільки вона працює швидше

Команда Rebuild будує заново весь проект (при запуску проекту клавішею F5 виконується команда Build, а не Rebuild). .